Novel Application of Genetic Sequencing algorithms to optimisation of hardware resource sharing for DSP

NOVEL APPLICATION OF GENETIC SEQUENCING ALGORITHMS TO OPTIMIZATION OF HARDWARE RESOURCE SHARING FOR DSP

S. McKeown and R. Woods+*

*CapnaDSP Ltd, Belfast
+ECIT Institute, Queens University Belfast

ABSTRACT

Field programmable gate array (FPGA) technology is a powerful platform for implementing computationally complex, digital signal processing (DSP) systems. Applications that are multi-modal, however, are designed for worse case conditions. In this paper, genetic sequencing techniques are applied to give a more sophisticated decomposition of the algorithmic variations, thus allowing an unified hardware architecture which gives a 10-25% area saving and 15% power saving for a digital radar receiver.

Index Terms— DSP, FFT, FPGA, sequencing, algorithm design

I. INTRODUCTION

Field programmable gate array (FPGA) implementation gains comes from creating an architecture which best matches algorithmic requirements by fixing the functionality. However, in some single application DSP systems, there is a need for several modes of operation [1] resulting in an over-engineered solution to meet worse case conditions unless dynamic reconfiguration is used but this is fraught with design and verification issues [2]; the convention and complexity in decomposing and analyzing optimal resource usage across several time domains makes alternatives problematic.

It is vital to identify resources from mutually exclusive functionality at the largest degree of common granularity. The difficulty lies in decomposing and ordering the resources so that the overhead involved in reusing them across the various modes of operation, does not dwarf the gains. Here we apply genetic sequencing namely the Smith-Waterman variation [3] to find locally optimal sequence alignments across multiple search spaces.

The paper is organized as follows: the need for multi-model systems are described in section II and an example given in section III; section IV outlines the design methodology; the application of the Smith-Waterman algorithm is described in section V and followed by the conclusions.

II. MULTI-MODAL DSP APPLICATIONS

Variable length DSP algorithms are widely used to match functionality to the operating requirements during run-time; the variable length FFT algorithm as part of an industrial digital receiver application is considered here as a point example indicative of the wider domain.

III. DIGITAL RADAR RECEIVER

In electronic warfare (EW) environments, radar systems are used to detect and classify various threats. They are data driven, flexible, process data at Gigasamples per second (GSPS) rates and can adapt their processing requirements as required. Many designers opt to use FPGAs as they combine both low NRE costs with the performance advantages.

III-A. DDR flexibility

In EW environments, there are a number of modes that need to be supported, e.g. entirely encapsulating a pulse in as short a window as possible for sensitivity and time of arrival (TOA). Ideally, this would involve running multiple overlapping FFT functions on a moving window, for a range of point sizes in real-time; this is impractical and so variable length cores based on the Cooley-Tukey algorithm are commonly used; these exploit the algorithmic structure to allow larger algorithms to be built out of smaller transforms, with an additional stage for increased length. It is shown in Fig. 1 how the 8 point transform can provide the functionality of 2 and 4 point transforms. The latter stages are only utilized when the full transform length is required.

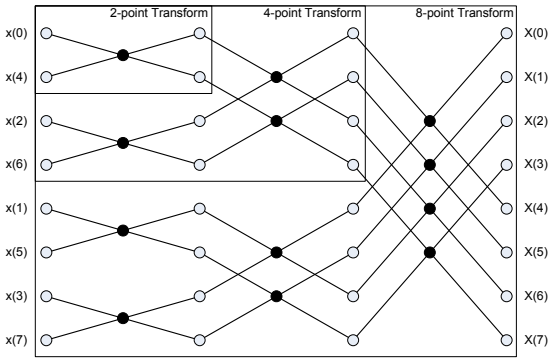


Fig. 1. FFT: algorithm structure

The time spent at each point size depends on the precise mission criteria and operating environment. This ultimately determines core resource utilization, since latter stages are redundant when processing smaller windows. Moreover,

unless the capability to turn these latter stages off is deliberately designed into the core, data is still processed causing unnecessary transitions. Examination of the data shows that the majority of pulse widths are relatively short, implying shorter length FFTs. In addition, there is a need to overlap FFT windows to avoid creation of spurious results and reducing TOA resolution.

IV. MULTI-MODAL SYSTEM ARCHITECTING

Creating a unified system architecture supporting multi-modal functionality can be considered by decomposing the individual algorithms into their constituent parts at a suitable level of granularity, analysing commonality between mutually exclusive operations of each mode across time domains and then creating the unified architecture to support the functional needs. By exploiting FPGA programmable resources, an unified architecture can be derived that supports use of resources in a time-shared manner.

IV-A. General Approach

Several algorithmic variations are mapped to a static architecture of underlying processing elements (PEs) with a flexible routing structure. The *reconfigurable mux* concept [4] is a powerful way of describing the combined modes of operation onto this flexible resource and allowing control of the decomposition granularity and flow of data across the modes of operation. Fig. 2 shows how this is applied to a real instance of 256pt and 16pt FFT processing; the first step applies the *mux* to describe mutually exclusive modes of operation by isolating the sub-graphs that are common.

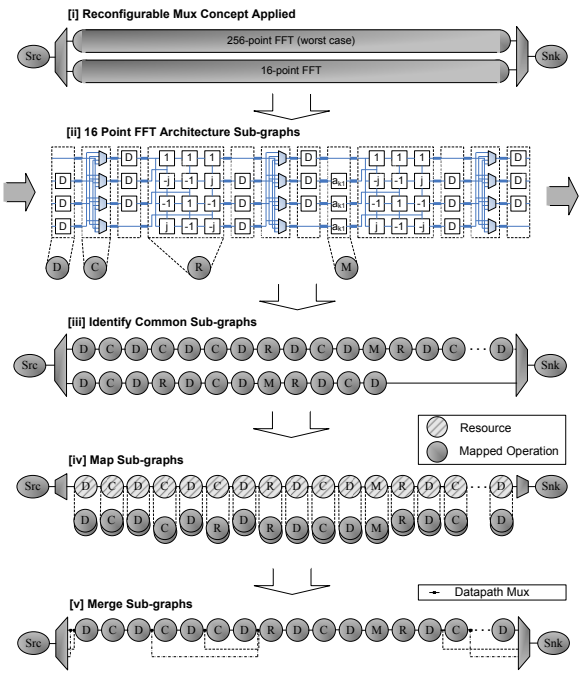


Fig. 2. FFT Algorithm: Datapath Configuration

In the second step, the design is examined for repetitive instances of strongly interconnected sub-graphis between the mutually exclusive functions. Sub-graphs are mapped as PEs in step 3, using the principles of set theory to maximize overlap in the intersection between mutually exclusive functions. The designer can optimize the design by adding additional select resources to achieve a balance between desired functionality and resource utilization. The mapped sub-graphs are merged in the final step, to produce a single dataflow graph (DFG) matching the resource requirements of the union of the individual functions. A flexible datapath routing structure using multiplexers is incorporated to provide mutually exclusive functions on demand.

V. SMITH-WATERMAN APPROACH

The Smith-Waterman algorithm is normally used to identify similar regions between two nucleotide protein sequences. For every sequence, the algorithm calculates all possible paths. Parameterizable gap penalties are used to find the highest match across areas of low similarity. In each search space iteration, there are four possible ways of forming and scoring a new segment path, 1) alignment of segment with residue of a remaining unmatched sequence, 2) deletion where the compared segment is matched to a gap and scored according to the size of the gap, 3) insertion where a gap is instead matched to a residue segment or, 4) where the search is terminated. Each option is given a score depending on weighted parameter values and the highest chosen as the result of any particular iteration.

The search matrix H is constructed as follows:

$$H(i,0) = 0, 0 \leq i \leq m \tag{1}$$

$$H(0,j) = 0, 0 \leq j \leq n \tag{2}$$

if $a_i = b_j w(a_i, b_j) = w(match)$ or $a_i ! = b_j w(a_i, b_j) = w(mismatch)$

$$H(i,j) \quad = \quad max, \begin{cases} 0 \\ H(i-1, j-1) + w(a_i, b_j) \\ H(i-1, j) + w(a_i, -) \\ H(i, j-1) + w(-, b_j), \end{cases} \tag{3}$$

for $Null, Match/Mismatch, Deletion, Insertion,$ with $1 \leq i \leq m, 1 \leq j \leq n.$

To obtain optimal local alignment the highest matrix value (i, j) is started with and backtracked until a cell with zero score is encountered giving the highest scoring local alignment; the alignment is then reconstructed and the process repeated. The simple sequences $'CAGCCUCGCUUAG'$ and $'AAUGCCAUUGACGG'$ are aligned as 'GCC–UCG' and 'GCCAUUG'.

V-A. Application of Smith-Waterman

The approach generates a decomposed sequence and identifies the largest degree of granularity onto which repetitive data local sub-graphs can be defined across algorithmic variations. The search space is prioritized, first in terms of data locality to ensure efficient communications between processing blocks, and secondly, in terms of achieving repetitive clusters common across algorithmic variations.

A data-flow modeling paradigm takes two or more data-flow graphs as input, compares them and merges overlapping functionality onto the same nodes, producing the combined DFG. The decomposition phase takes a description of the type in Fig. 3(a), for each operation specified in the system functionality requirements definition. From this, a high level decomposition is produced with functionality represented by a linear string sequence allowing clustering of the nodes as strongly interconnected sub-graphs, preserving data locality and maximization of the size of high level blocks common across algorithmic variations.

Within each DFG, nodes are assigned a value, ρ, which represents the degree to which their operation is dependent on external data sources and sinks.

$$\rho = \Sigma(\text{elements from unique peers}) + \Sigma(\text{elements to unique peers})$$

For example, the multiplexer node, labeled 'mux', has 4 inputs and 1 output so has a value ρ of 5 (Fig. 3(b)). Where token sizes are not uniform, this can be calculated as a weighted value. The ρ values are then arranged into an array (Fig. 4[i]) according to their relative positions in the processing chain.

Nodes are grouped to minimize the input/output needs across clusters. A pipelined processor lends itself to a column based clustering approach, but a wider search space could be used if required. Grouping column-wise, the individual ρ values are replaced by the value for the column as a whole (Fig. 4[ii]).

Each unique column is assigned a letter representing its functionality, enabling the functionality of algorithmic variations to be described as a string. Algorithm string sequences are then analyzed to detect shared sequences whose elements only appear in that order (Fig. 4[iii]). Within these unique sub-sequences, columns are clustered to minimize ρ and optimize cluster size (Fig. 4[iv]). Algorithms are then described in terms of these new clusters, the outcome of which is an algorithmic description; this functionality is described as a sequence of processing nodes common across algorithmic variations (Fig. 4[v]). These string sequences are passed to the mapping and merger phase.

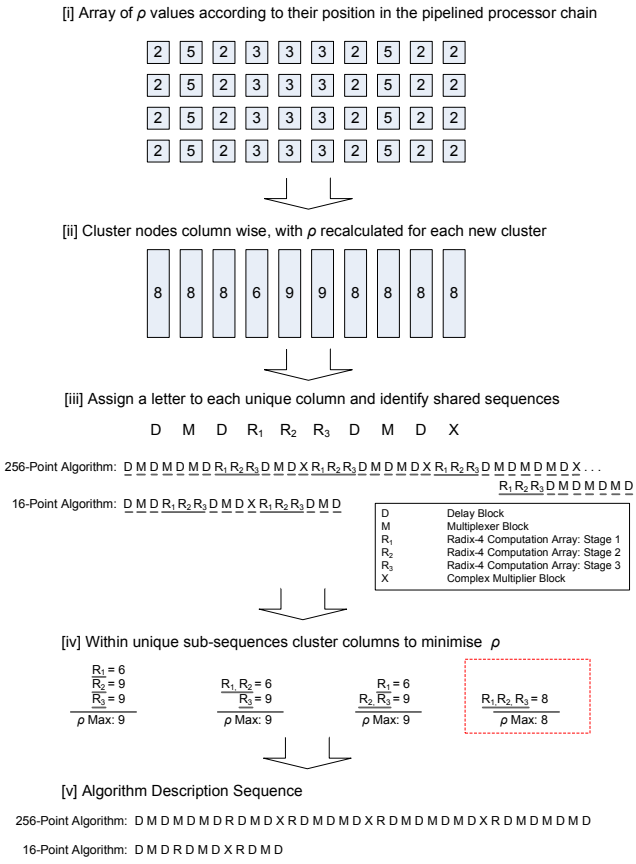


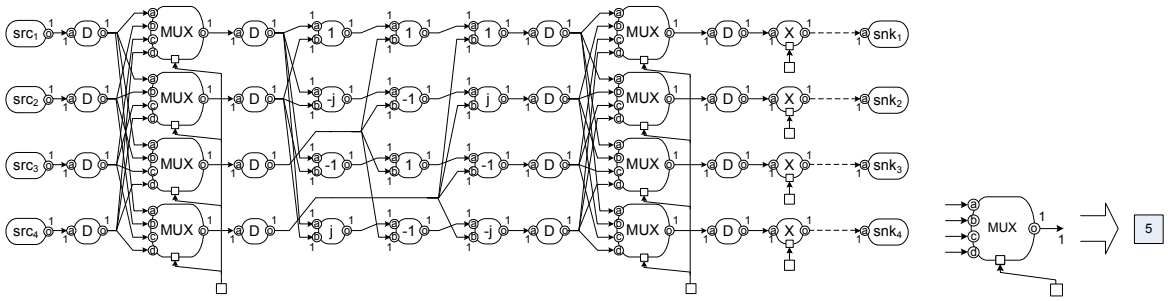
Fig. 4. Decomposition: FFT example

V-B. Smith-Waterman Mapping

A string sequence is formed for functionality required in each mutually exclusive mode of operation and are compared using Smith-Waterman over several iterations, until all functionality is mapped. A flexible routing structure is then specified to implement the required functionality. This process can be run several times to produce a range of implementation options, describing achievable functionality and resource requirements for each option optimal to user defined penalty parameters.

Fig. 5 illustrates this approach using the algorithmic sequences generated by the decomposition process. A string sequence is first formed which represents the minimum algorithmic needs of each mutually exclusive mode of operation, a 256-point and a 16-point transform (Fig. 5[i]). In addition to these base line requirements, extra functionality is added, in the form of an additional transform in 16-point mode (Fig. 5[ii]). Functionality is mapped over the the first iteration of the algorithm, giving an initial similarity measure. A new sequence is then formed representing unmapped resources from the first iteration. This process is repeated until all resources are mapped or the similarity measure reaches zero, in which case additional resources are required to complete the mapping (Fig. 5[iii]).

Interconnect needs are implicitly defined in the mapping stage which ensures configurable datapath requirements are minimized, but when the similarity measure between se-



(a) Core Library DFG: FFT (b) Multiplexer

Fig. 3. Multiplexer

quences is low, a trade-off can be made between additional resources and interconnect by setting the similarity measure cut-off point to a value greater than zero. Replacing the string sequence letters with the equivalent DFG actors, Fig. 5[iv] shows how functionality is mapped to actors, with different shadings indicating at which iteration each actor is mapped. The functionality can then be alternated between the various modes of operation simply by changing the multiplexer's control signal.

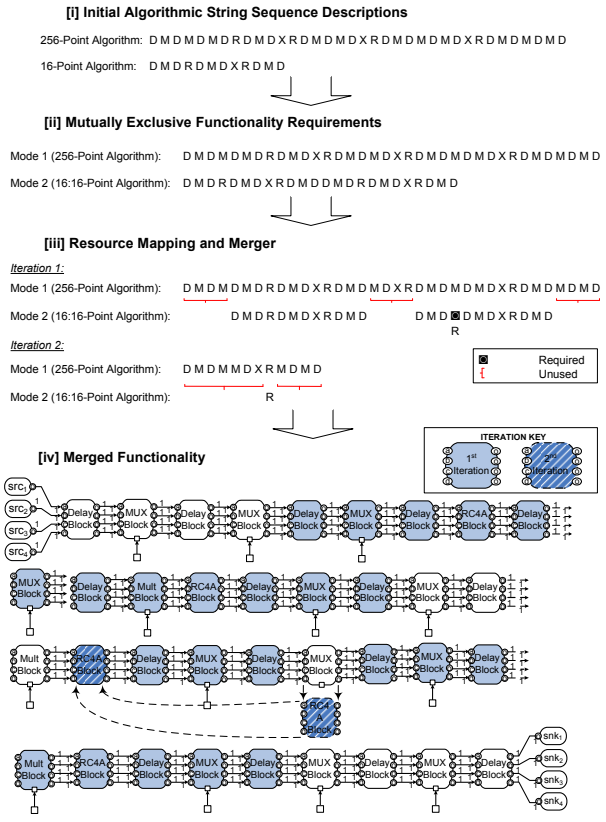


Fig. 5. Mapping and Merger: FFT example

V-C. Digital Radar Receiver Results

The technique described was applied to a digital receiver and benchmarked operating at 800MSPS for a 256pt and 2x16pt operation. The resulting multi-modal architecture requires 29 static and only 6 configurable datapaths between PEs and performs either one 256pt or two 16pt transforms. However, this is achieved at significant resource and power savings as shown in Table. I. The new architecture (QFFT_R) uses 89.7% slices and 75% of the multipliers of the benchmark (QFFT_2), and outperforms the benchmark in measured power consumption (84%) for two captured real radar data-sets on a Xilinx Virtex-2 device.

Core		Resources			Power Consumption (mW)	
Name	Functionality	Slices	LUTs	Mult 18x18	Data Set 1	Data Set 2
QFFT_R Created Architecture	256/16-16	**5794** (89.7%)	**7917** (88.9%)	**27** (75.0%)	**2132** (84.3%)	**1998** (84.5%)
QFFT_2 Benchmark	256/16-16	6461	8907	36	*2530*	*2365*

Table I. Power and Resource Figures

VI. CONCLUSIONS

Genetic sequencing has been applied as a means to maximize the functionality of a fixed resource to better meet user requirements. A 10-25% area saving and 15% power saving has been achieved over a previously optimised solution.

VII. REFERENCES

[1] T. Tanaka, Y. Hirasawa, and Y. Yamashita, "Variable-length lapped transforms with a combination of multiple synthesis filter banks for image coding," *IEEE Transactions on Image Proc.*, vol. 15, no. 1, pp. 81–88, Jan. 2006.

[2] R. Pandey and M. Bushnell, "Architecture for variable-length combined fft, dct, and mwt transform hardware for a multi-modewireless system," in *Proc. Int. Conf. Embedded Systems VLSI Design*, Jan. 2007, pp. 121–126.

[3] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. of Molecular Biology*, 1981.

[4] N. Shirazi, W. Luk, and P. Cheung, "Automating production of run-time reconfigurable designs," in *IEEE Proc. on FCCM*, Apr. 1998, pp. 147–156.